# Accelerated stochastic multiplicative update with gradient averaging for nonnegative matrix factorizations

Hiroyuki Kasai
*Graduate School of Informatics and Engineering*
*The University of Electro-Communications*
Tokyo, Japan
kasai@is.uec.ac.jp

***Abstract*—Nonnegative matrix factorization (NMF) is a powerful tool in data analysis by discovering latent features and part-based patterns from high-dimensional data, and is a special case in which factor matrices have low-rank nonnegative constraints. Applying NMF into huge-size matrices, we specifically address stochastic multiplicative update (MU) rule, which is the most popular, but which has slow convergence property. This present paper introduces a gradient averaging technique of stochastic gradient on the stochastic MU rule, and proposes an accelerated stochastic multiplicative update rule: SAGMU. Extensive computational experiments using both synthetic and real-world datasets demonstrate the effectiveness of SAGMU.**

***Index Terms*—nonnegative matrix factorization, multiplicative update, stochastic gradient, gradient averaging**

## I. Introduction

Nonnegative matrix factorization (NMF) is a fundamental problem for discovering nonnegative latent factors and/or performing dimensionality reduction. NMF has been successfully applied in diverse technical fields, such as pattern recognition, image/video analysis, text mining, bioinformatics and Web analysis because non-negativity of the obtained factors gives understandable interpretations of data of interest. NMF approximates a nonnegative matrix $\mathbf{V}$ as a product of two nonnegative matrices $\mathbf{W}$ and $\mathbf{H}$. More concretely, given $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, NMF seeks a factorization of the form

$$\mathbf{V} \approx \mathbf{WH},$$

where $\mathbf{W} \in \mathbb{R}_+^{F \times K}$ and $\mathbf{H} \in \mathbb{R}_+^{K \times N}$ are nonnegative *factor matrices*. $K$ is usually chosen such that $K \ll \min\{F, N\}$, that is, $\mathbf{V}$ is approximately represented by the two *low-rank* matrices. Note that, throughout the paper, we denote scalars by lower-case letters $(a, b, c, \ldots)$, vectors as bold lower-case letters $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots)$, and matrices as bold-face capitals $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots)$. An element at $(i, j)$ of a matrix $\mathbf{A}$ is represented as $[\mathbf{A}]_{i,j}$.

This problem is formulated as a constrained minimization problem in terms of the *Euclidean distance* as

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & \frac{1}{2}\|\mathbf{V} - \mathbf{WH}\|_F^2, \\ \text{s.t.} \quad & [\mathbf{W}]_{f,k} \geq 0, [\mathbf{H}]_{k,n} \geq 0, \quad \forall f, n, k, \end{aligned} \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm. Because Problem (1) is a *non-convex* optimization problem, finding its global minimum is NP-hard. For this problem, Lee and Seung proposed a simple but effective calculation algorithm [1] as

$$\begin{aligned} \mathbf{H} &\leftarrow \mathbf{H} \odot \frac{\mathbf{W}^T\mathbf{V}}{\mathbf{W}^T\mathbf{WH}}, \\ \mathbf{W} &\leftarrow \mathbf{W} \odot \frac{\mathbf{VH}^T}{\mathbf{WHH}^T}, \end{aligned} \tag{2}$$

where $\odot$ (resp. $\frac{\cdot}{\cdot}$) denotes the component-wise product (resp. division) of matrices, which finds a *local optimal* solution of Problem (1). This rule is designated as the *multiplicative update* (MU) rule because a new estimate is represented as the product of a current estimate and some factor. The *global convergence to a stationary point* is guaranteed under slightly modified update rules or constraints [2], [3]. Nevertheless, many efficient algorithms have been developed because the MU rule is hindered by a slow convergence [4]–[7].

Considering big data processing, an *online* or *stochastic optimization* algorithm alleviates high computational burden and memory consumption. Designating the algorithms mentioned earlier as *batch NMF*, *online NMF* and *stochastic NMF* have gained much attention recently. They specifically consider the situation where one column of $\boldsymbol{v}_n \in \mathbf{V}$ $(n \in [N])$ is fed into the algorithm every iteration, and $\boldsymbol{h}_n$ and $\mathbf{W}$ are updated, where $\mathbf{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]$ and $\mathbf{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N]$. For this particular case, they specifically consider an equivalent reformulation of Problem (1) as

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & \sum_{n=1}^{N} \frac{1}{2}\|\boldsymbol{v}_n - \mathbf{W}\boldsymbol{h}_n\|_2^2, \\ \text{s.t.} \quad & [\mathbf{W}]_{f,k} \geq 0, [\mathbf{H}]_{k,n} \geq 0, \quad \forall f, n, k. \end{aligned} \tag{3}$$

We have several studies [8]–[12] and its robust variant [13] in literature. However, they still exhibit a slow convergence.

The *stochastic gradient descent* (SGD) algorithm [14] has become the method of choice for solving big data optimization problems. Although it is beneficial because of the low and constant cost per iteration independent of $N$, the *convergence*

ISBN 978-90-827970-1-5 

*rate* of SGD is also slower than that of full GD even for strongly-convex cases [15]. For this issue, various *variance reduction* (VR) approaches that have been proposed recently have achieved superior convergence rates in (strongly-) convex and non-convex functions. Furthermore, very recently, SVRMU has been proposed to accelerate the convergence speed of ONMF [13] by exploiting the VR technique [16] specialized for the MU rule.

This paper presents a proposal of a novel *accelerated* stochastic multiplicative update with a gradient averaging technique: SAGMU. The proposed SAGMU outperforms the SVRMU and other existing algorithms as seen later. This paper is organized as follows. Section II briefly introduces some state-of-the-art variance reduction algorithms in stochastic optimization algorithms. Section III proposes the SAGMU algorithm, and Section IV explains extensions of SAGMU to the accelerated variant (SAGMU-ACC), and the robust variant for outliers (R-SAGMU). The convergence analysis is given in Section V. Exhaustive comparisons in Section VI suggest that the proposed SAGMU algorithms robustly outperform state-of-the-art algorithms across different synthetic and real-world datasets. It is noteworthy that the discussion presented here is applicable to other distance functions than the Euclidean distance.

## II. Variance reduction algorithms in stochastic optimization

An algorithm designated to solve Problem (1) without nonnegative constraints is begin eagerly sought in the machine learning field. When designating $\mathbf{W}$ and $\mathbf{H}$ as $w$, and designating the $n$-th ($n \in [N]$) inner term of Problem (3) as $f_n(w)$, respectively, the *full gradient decent* (GD) algorithm is the most straightforward approach as

$$w_{t+1} = w_t - \eta g_t,$$

where $g_t$ is $\nabla f(w_t) = \sum_{n=1}^{N} \nabla f_n(w_t)$ and $\eta$ is stepsize. However, the calculation of $\nabla f(w_t)$ is expensive especially when $N$ is extremely large. A popular and effective alternative is SGD, which sets $g_t$ at $t$ as $\nabla f_{n_t}(w_t)$ for the $n_t$-th ($n_t \in [N]$) sample that is selected uniformly at random. More specifically, SGD updates $w_t$ as

$$w_{t+1} = w_t - \eta \nabla f_{n_t}(w_t),$$

and assumes an *unbiased estimator* of the full gradient as $\mathbb{E}_{n_t}[\nabla f_{n_t}(w_t)] = \nabla f(w_t)$. Apparently, the calculation cost per iteration is independent of $N$. *Mini-batch* SGD uses $g_t = 1/|\mathcal{S}_t| \sum_{n_t \in \mathcal{S}_t} \nabla f_{n_t}(w_t)$, where $\mathcal{S}_t$ is the set of samples of size $|\mathcal{S}_t|$. However, because SGD requires a *diminishing* stepsize algorithm to guarantee the convergence, SGD suffers from a slow convergence rate.

To accelerate this rate, the variance reduction (VR) techniques [17]–[22] explicitly or implicitly exploit a full gradient estimation to reduce the variance of noisy stochastic gradient, leading to superior convergence properties. We can regard this approach as a hybrid algorithm of GD and SGD. A representative research among them is the stochastic variance reduced gradient (SVRG) algorithm [17]. SVRG has a double-loop structure, i.e., the inner loop indexed by $t$ and the outer loop (i.e., epoch) indexed by $s$. SVRG reduces the variance of noisy stochastic gradients in the inner loop by exploiting the full gradient that is periodically calculated at the outer loop. The detailed algorithm of SVRG is as follows: it stores $\tilde{w} = w_t^{s-1}$ indexed by $t = 0, \cdots, m_{s-1} - 1$ at the end of the previous $(s-1)$-th epoch, where $m_{s-1}$ is the number of the inner iterations of the $(s-1)$-th epoch. It also sets the initial value of the inner loop in the $s$-th epoch as $w_0^s = \tilde{w}$. In parallel, the full gradient $\nabla f(\tilde{w})$ at $\tilde{w}$ is computed and stored. Meanwhile, every inner iteration, it randomly selects $n_t^s \in [N]$ for each $\{t, s\} \geq 0$, and computes a *modified stochastic gradient* $g_t^s$ as

$$g_t^s = \nabla f_{n_t^s}(w_t^s) - \nabla f_{n_t^s}(\tilde{w}^s) + \nabla f(\tilde{w}^s).$$

It should be noted that the calculated $g_t^s$ is also an unbiased estimator of the full gradient. SVRG enjoys a linear convergence rate for smooth and strongly-convex functions. Very recently, by extending the strategy of SVRG into the MU rule, SVRMU has been proposed in [16]. It exhibits superior performances on convergence speeds and on some image processing applications.

The stochastic average gradient (SAG) algorithm [18] is another representative study for the VR technique *without* relying on the double-loop structure as SVRG. For the selected $n_t \in [N]$ for each $t \geq 0$, SAG stores the *most recently computed* gradients other than $\nabla f_{n_t}(w_t)$, and calculates the modified stochastic gradient as

$$g_t = \frac{1}{N}\left[\nabla f_{n_t}(w_t) - \nabla f_{n_t}(w_{t_{n_t}}) + \sum_n \nabla f_n(w_{t_n})\right],$$

where $t_{n_t} < t$ is the iteration in which $\nabla f_{n_t}(w_{t_{n_t}})$ was the most recently evaluated gradient for the $n_t$-th sample at $w_{t_{n_t}}$. Although it needs additional memories, SAG competes effectively and sometimes outperforms state-of-the-art stochastic gradient algorithms including SVRG. Therefore, this present paper exploits the fundamental approach of SAG for the MU rule, and demonstrates superior performances against SVRMU and others.

## III. Stochastic average gradient multiplicative update (SAGMU)

After a brief introduction of the *stochastic multiplicative update* (SMU) algorithm introduced in [16], this section details the proposed stochastic average gradient MU algorithm, i.e., SAGMU.

The problem setting is the following: we assume that the $n_t$-th column of $\mathbf{V}$, i.e. $\boldsymbol{v}_{n_t}$, is selected at the $t$-th iteration uniformly at random. Then, $\boldsymbol{h}_{n_t}$ and $\mathbf{W}$ are updated by extending (2) as

$$\boldsymbol{h}_{n_t} \leftarrow \boldsymbol{h}_{n_t} \odot \frac{\mathbf{W}^T \boldsymbol{v}_{n_t}}{\mathbf{W}^T \mathbf{W} \boldsymbol{h}_{n_t}},$$
$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\boldsymbol{v}_{n_t} \boldsymbol{h}_{n_t}^T}{\mathbf{W} \boldsymbol{h}_{n_t} \boldsymbol{h}_{n_t}^T}. \tag{4}$$

ISBN 978-90-827970-1-5 © EURASIP 2018

Especially, the MU rule of $\mathbf{W}$ is regarded as a special case of SGD as presented below.

$$\mathbf{W} \leftarrow \mathbf{W} - \mathbf{S}^t \odot (\mathbf{W}\boldsymbol{h}_{n_t}\boldsymbol{h}_{n_t}^T - \boldsymbol{v}_{n_t}\boldsymbol{h}_{n_t}^T),$$

where $\mathbf{S}^t \in \mathbb{R}_+^{F\times K}$ is an adaptive stepsize of *matrix form* of

$$\mathbf{S}^t = \frac{\alpha \mathbf{W}}{\mathbf{W}\boldsymbol{h}_{n_t}\boldsymbol{h}_{n_t}^T}.$$

where $0 < \alpha \leq 1$ is the *stepsize ratio* that ensures that $\mathbf{W}$ and $\mathbf{H}$ are nonnegative when those initial values are nonnegative. The case of $\alpha = 1$ produces (4) exactly [16].

Based on this interpretation, we consider a gradient averaging algorithm for SMU. Here, we first designate $t_{n_t}$ for the iteration index $t$ where the $n_t$-th sample, i.e., $\boldsymbol{v}_{n_t}$, is most recently processed. Also, $\mathbf{W}$ and $\boldsymbol{h}_{n_t}$ calculated at $t_{n_t}$ are denoted as $\mathbf{W}^{t_{n_t}}$ and $\boldsymbol{h}_{n_t}^{t_{n_t}}$, respectively. Similarly as SAG, SAGMU keeps the most recently computed gradients other than $\nabla f_{n_t}(w_t)$, and calculates the modified gradient using them. More specifically, using $m$ instead of $n_t$ for notation simplicity, we first randomly select $m$ and update $\boldsymbol{h}_m^t$ by foloowing (4) as

$$\boldsymbol{h}_m^t = \boldsymbol{h}_m^{t_m} \odot \frac{(\mathbf{W}^t)^T \boldsymbol{v}_m}{(\mathbf{W}^t)^T \mathbf{W}^t \boldsymbol{h}_m^{t_m}},$$

for each $t \geq 0$. Then, we update $\mathbf{W}^t$ into $\mathbf{W}^{t+1}$ at $t$ with an appropriate stepsize $\mathbf{S}^t$ as shown below.

$$\begin{aligned}
\mathbf{W}^{t+1} &= \mathbf{W}^t - \frac{\mathbf{S}^t}{N} \odot \Bigg[\sum_{n=1}^{N}(\mathbf{W}^{t_n}\boldsymbol{h}_n^{t_n}(\boldsymbol{h}_n^{t_n})^T - \boldsymbol{v}_i(\boldsymbol{h}_n^{t_n})^T) \\
&\quad - (\mathbf{W}^{t_m}\boldsymbol{h}_m^{t_m}(\boldsymbol{h}_m^{t_m})^T - \boldsymbol{v}_m(\boldsymbol{h}_m^{t_m})^T) \\
&\quad + \mathbf{W}^t\boldsymbol{h}_m^t(\boldsymbol{h}_m^t)^T - \boldsymbol{v}_m(\boldsymbol{h}_m^t)^T\Bigg] \\
&= \mathbf{W}^t - \mathbf{S}^t \odot \Bigg[ \\
&\underbrace{\frac{1}{N}\Bigg(\overbrace{\sum_{n=1}^{N}\mathbf{W}^{t_n}\boldsymbol{h}_n^{t_n}(\boldsymbol{h}_n^{t_n})^T}^{\mathbf{F}_{\mathrm{sum}}} + \overbrace{\boldsymbol{v}_m(\boldsymbol{h}_m^{t_m})^T}^{\mathbf{G}(m)} + \mathbf{W}^t\boldsymbol{h}_m^t(\boldsymbol{h}_m^t)^T\Bigg)}_{\mathbf{Q}^t} \\
&\underbrace{- \frac{1}{N}\Bigg(\overbrace{\sum_{n=1}^{N}\boldsymbol{v}_i(\boldsymbol{h}_n^{t_n})^T}^{\mathbf{G}_{\mathrm{sum}}} + \overbrace{\mathbf{W}^{t_m}\boldsymbol{h}_m^{t_m}(\boldsymbol{h}_m^{t_m})^T}^{\mathbf{F}(m)} + \boldsymbol{v}_m(\boldsymbol{h}_m^t)^T\Bigg)\Bigg]}_{\mathbf{P}^t}. \quad (5)
\end{aligned}$$

Here, we respectively denote the first and the second term multiplied with $1/N$ in the squared bracket of (5) as $\mathbf{Q}^t \in \mathbb{R}_+^{F\times K}$ and $\mathbf{P}^t \in \mathbb{R}_+^{F\times K}$. When $\mathbf{S}^t = \alpha^t\mathbf{W}^t/\mathbf{Q}^t$ with the stepsize ratio $\alpha^t$, the update rule in (5) is reformulated as presented below.

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \frac{\alpha^t \mathbf{W}^t}{\mathbf{Q}^t} \odot (\mathbf{Q}^t - \mathbf{P}^t). \quad (6)$$

For an efficient practical algorithm, we store $\mathbf{F}_{\mathrm{sum}}$ and $\mathbf{G}_{\mathrm{sum}}$ defined as

$$\mathbf{F}_{\mathrm{sum}} = \sum_{n=1}^{N}\mathbf{W}^{t_n}\boldsymbol{h}_n^{t_n}(\boldsymbol{h}_n^{t_n})^T, \quad \mathbf{G}_{\mathrm{sum}} = \sum_{n=1}^{N}\boldsymbol{v}_i(\boldsymbol{h}_n^{t_n})^T.$$

Also, the most recently calculated $\mathbf{W}^{t_m}\boldsymbol{h}_m^{t_m}(\boldsymbol{h}_m^{t_m})^T$ and $\boldsymbol{v}_m(\boldsymbol{h}_m^{t_m})^T$ for the $m$-th sample are stored as

$$\mathbf{F}(m) = \mathbf{W}^{t_m}\boldsymbol{h}_m^{t_m}(\boldsymbol{h}_m^{t_m})^T, \quad \mathbf{G}(m) = \boldsymbol{v}_m(\boldsymbol{h}_m^{t_m})^T.$$

The overall algorithm is summarized in Algorithm 1. It should be noted that the mini-batch variant of Algorithm 1 is straightforward.

## IV. Variants of SAGMU

This section introduces two variants of SAGMU, which are an accelerated variant of SAGMU (SAGMU-ACC) and a robust variant of SAGMU (R-SAGMU).

### A. Accelerated SAGMU (SAGMU-ACC)

The acceleration technique proposed in [16] can be also applicable to SAGMU. We can repeat the calculation of $\boldsymbol{h}_m$ several times, which corresponds to Step 4 in Algorithm 1, before the computation of $\mathbf{W}^t$. We call this variant as SAGMU-ACC. The stopping condition is different from that of [16], but the details are omitted.

### B. Robust SAGMU (R-SAGMU)

The outlier in $\mathbf{V}$ causes dreadful degradation of the approximation of $\mathbf{V}$. To address this issue, robust batch-NMF [23] and robust online-NMF [13], [16] have been proposed. This variant, R-SAGMU, also handles the same problem within the SAGMU framework. Given the outlier matrix $\mathbf{R} = [\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N] \in \mathbb{R}_+^{F\times N}$, R-SAGMU seeks $\mathbf{V} \approx \mathbf{W}\mathbf{H} + \mathbf{R}$, of which problem is reformulated from Problem (3) as

$$\begin{aligned}
\min_{\mathbf{W},\mathbf{H},\mathbf{R}} \quad & \sum_{n=1}^{N}\frac{1}{2}\|\boldsymbol{v}_n - \mathbf{W}\boldsymbol{h}_n - \boldsymbol{r}_n\|_2^2 + \lambda\|\boldsymbol{r}_n\|_1, \\
\text{s.t.} \quad & [\mathbf{W}]_{f,k} \geq 0,\ \boldsymbol{h}_n \geq 0,\ \boldsymbol{r}_n \geq 0, \quad \forall f, n, k,
\end{aligned}$$

**Algorithm 1** Stochastic average gradient multiplicative update (SAGMU)

**Require:** $\mathbf{V}$.

1: Initialize $\mathbf{F}_{\mathrm{sum}} = \mathbf{G}_{\mathrm{sum}} = \mathbf{0}$ and $\mathbf{F}(n) = \mathbf{G}(n) = \mathbf{0}$ for all $n \in [N]$.
2: **for** $t = 0, 1, \ldots$ **do**
3: Choose $m = n_t \in [N]$ uniformly at random.
4: Update $\boldsymbol{h}_m^t = \boldsymbol{h}_m^{t_m} \odot ((\mathbf{W}^t)^T\boldsymbol{v}_m)/((\mathbf{W}^t)^T\mathbf{W}^t\boldsymbol{h}_m^{t_m})$.
5: Calculate $\mathbf{Q}^t = \frac{1}{N}(\mathbf{F}_{\mathrm{sum}} + \mathbf{G}(m) + \mathbf{W}^t\boldsymbol{h}_m^t(\boldsymbol{h}_m^t)^T)$.
6: Calculate $\mathbf{P}^t = \frac{1}{N}(\mathbf{G}_{\mathrm{sum}} + \mathbf{F}(m) + \boldsymbol{v}_m(\boldsymbol{h}_m^t)^T)$.
7: Calculate the stepsize ratio $\alpha^t$.
8: Update $\mathbf{W}^{t+1} = \mathbf{W}^t - \alpha^t\mathbf{W}^t/\mathbf{Q}^t \odot (\mathbf{Q}^t - \mathbf{P}^t)$.
9: Update $\mathbf{F}_{\mathrm{sum}} = \mathbf{F}_{\mathrm{sum}} + \mathbf{W}^t\boldsymbol{h}_m^t(\boldsymbol{h}_m^t)^T - \mathbf{F}(m)$.
10: Update $\mathbf{G}_{\mathrm{sum}} = \mathbf{G}_{\mathrm{sum}} + \boldsymbol{v}_m(\boldsymbol{h}_m^t)^T - \mathbf{G}(m)$.
11: Update $\mathbf{F}(m) = \mathbf{W}^t\boldsymbol{h}_m^t(\boldsymbol{h}_m^t)^T$.
12: Update $\mathbf{G}(m) = \boldsymbol{v}_m(\boldsymbol{h}_m^t)^T$.
13: **end for**

ISBN 978-90-827970-1-5 © EURASIP 2018

where $\lambda > 0$ is the regularization parameter, and $\|\cdot\|_1$ is the $\ell_1$-norm. For this problem, the update rule (5) is redefined as

$$\begin{aligned}\mathbf{W}^{t+1} &= \mathbf{W}^t - \frac{\mathbf{S}^t}{N} \odot \Bigg[ \Big( \sum_{n=1}^{N} (\mathbf{W}^{t_n} \boldsymbol{h}_n^{t_n} + \boldsymbol{r}_n^{t_n})(\boldsymbol{h}_n^{t_n})^T \\ &\quad + \boldsymbol{v}_m (\boldsymbol{h}_m^{t_m})^T + (\mathbf{W}^t \boldsymbol{h}_m^t + \boldsymbol{r}_m^t)(\boldsymbol{h}_m^t)^T \Big) \\ &\quad - \Big( \sum_{n=1}^{N} \boldsymbol{v}_i (\boldsymbol{h}_n^{t_n})^T + (\mathbf{W}^{t_m} \boldsymbol{h}_m^{t_m} + \boldsymbol{r}_m^{t_m})(\boldsymbol{h}_m^{t_m})^T \\ &\quad + \boldsymbol{v}_m (\boldsymbol{h}_m^t)^T \Big) \Bigg]. \end{aligned} \tag{7}$$

Accordingly, we respectively calculate the following:

$$\begin{aligned} \boldsymbol{h}_m^t &\leftarrow \boldsymbol{h}_m^{t_m} \odot \frac{(\mathbf{W}^t)^T \boldsymbol{v}_m}{(\mathbf{W}^t)^T \mathbf{W}^t \boldsymbol{h}_m^{t_m} + (\mathbf{W}^t)^T \boldsymbol{r}_m^{t_m}}, \\ \boldsymbol{r}_m^t &\leftarrow \boldsymbol{r}_m^{t_m} \odot \frac{\boldsymbol{v}_m}{\mathbf{W}^t \boldsymbol{h}_m^{t_m} + \boldsymbol{r}_m^{t_m} + \boldsymbol{\lambda}_{F\times 1}}, \end{aligned}$$

where $\boldsymbol{\lambda}_{F\times 1} \in \mathbb{R}^F$ is the vector with all entries equal to $\lambda$.

## V. Convergence Analysis

The convergence analysis is similar to [24], [25] and follows exactly the same as that of [16]. Therefore, we omit the details, and introduce the result only.

**Theorem V.1.** *Define* $f_N(\boldsymbol{h}_n, \mathbf{W}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{2}\|\boldsymbol{v}_n - \mathbf{W}\boldsymbol{h}_n\|_2^2$, *and* $\hat{f}_N(\mathbf{W}) := \frac{1}{N}\sum_{n=1}^{N} l(\hat{\boldsymbol{h}}_n, \mathbf{W})$, *where* $\hat{\boldsymbol{h}}_n$ *is already calculated during the previous steps. Consider* $f(\boldsymbol{h}_n, \mathbf{W}) := \mathbb{E}_{\boldsymbol{v}}[l(\boldsymbol{h}_n, \mathbf{W})] = \lim_{N\to\infty} f_N(\boldsymbol{h}_n, \mathbf{W})$. *Assume that* $\{\boldsymbol{v}\}_{n=1}^{\infty}$ *are i.i.d. random processes, and bounded. Iterates of* $\mathbf{W}^t$ *for* $0 \le t$ *are compact. The initial* $\mathbf{W}^0$ *is nonnegative and has a full column rank.* $\hat{f}_N(\mathbf{W})$ *is positive definite and strictly convex.* $\alpha^t$ *generates a diminishing stepsize of* $\mathbf{S}^t$. *Then, the iterates* $\mathbf{W}^t$ *produced by Algorithm 1 asymptotically coincide with the stationary points of the minimization problem of* $f(\boldsymbol{h}_n, \mathbf{W})$.

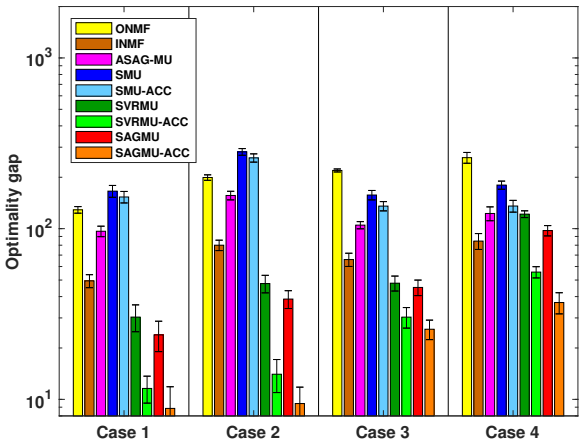
## VI. Numerical Experiments

We evaluate the effectiveness of SAGMU by comparing with state-of-the-art online and batch algorithms for NMF. We implemented all of the algorithms in MATLAB[1].

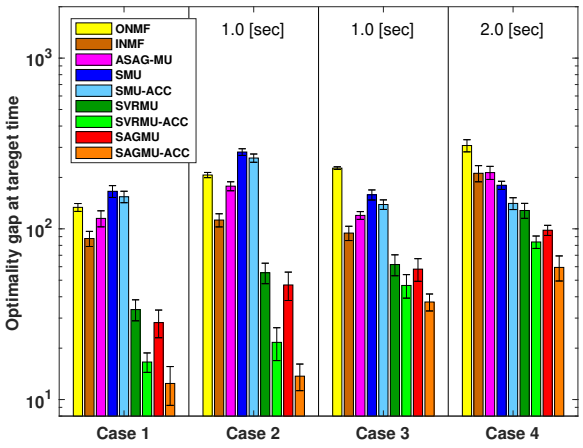
### A. Convergence behavior under clean synthetic data

We first evaluate convergence behaviors of SAGMU using synthetic datasets without outlier. For the synthetic data generation, the element $[\mathbf{W}_o]_{f,n}$ of the ground-truth $\mathbf{W}_o \in \mathbb{R}_+^{F\times K_o}$ is generated from a Gaussian distribution with a mean of zero and variance $1/\sqrt{K_o}$ for any $(f, n)$, where $K_o$ is the ground-truth rank dimension. Similarly, we generate $\mathbf{H}_o \in \mathbb{R}_+^{K_o \times N}$. Then, the clean data $\mathbf{V}_o$ are created as $\mathbf{V}_o = \mathcal{P}_{\mathcal{V}}(\mathbf{W}_o\mathbf{V}_o)$, where $\mathcal{V} = [0, 1]^{F\times N}$, and where $\mathcal{P}_{\mathcal{V}}$ is the normalization projector [13]. We set four parameters $(F, N, K_o, b)$, where $b$ is the batch size, as **Case 1:**$(300, 1000, 10, 100)$, **Case 2:**$(500, 1000, 10, 100)$, **Case 3:**$(300, 1000, 30, 100)$, and **Case 4:**$(300, 5000, 10, 500)$. The maximum epoch is 500. The following state-of-the-art methods are used for comparison: online MU (ONMF) [13], incremental MU (INMF) [8], ASAG-MU [11], SMU, SMU-ACC, SVRMU and SVRMU-ACC [16]. Our proposed algorithms include SAGMU and its accelerated variant SAGMU-ACC in Section IV.

[1]https://github.com/hiroyuki-kasai




(a) Optimality gap at final epoch




(b) Optimality gap at target time

Fig. 1: Optimality gap on synthetic dataset.

Figure 1 portrays results of *functional optimality gap*, of which optimal value is calculated using HALS [5] in advance. Figures 1:(a) and (b) present the optimality gap at the final epoch and at the *predefined* target time, respectively. The target times for each case are indicated at the top of Figure 1:(b). Overall, the proposed SAGMU(-ACC) outperforms other algorithms. Although INMF shows lower optimality gap than SAGMU does at **Case 4** in Figure 1:(a), we can see much better performances of SAGMU in terms of processing time from Figure 1:(b).

ISBN 978-90-827970-1-5 © EURASIP 2018

### B. Denoising and clustering of images with outlier

Using a real-world dataset, we next evaluate the performances of SAGMU in image processing applications. We use the CMU PIE dataset[2], which contains 337 subjects, captured under 15 view points and 19 illumination conditions. The maximum level of the pixel values is set to 50. All pixel values are normalized. We also randomly add entry-wise nonnegative outliers with density $\rho = 0.9$. All outliers are drawn from the i.i.d. from a uniform distribution $\mathcal{U}[30, 50]$. $K_o$ is fixed to 24. The methods of comparison include ONMF and its robust variant: R-ONMF [13], and the accelerated variant of R-SVRMU. The batch-based variant of R-ONMF (R-NMF) is also evaluated. We use the *peak signal-to-noise ratio* (PSNR) for the image denoising, which is calculated as $10 \log 10(V_{max}^2/\mathrm{MSE})$, where $V_{max}$ is the maximum value of the pixels, and $\mathrm{MSE} = \|\mathbf{V}_o - \mathbf{WH}\|_F^2/(FN)$. The *normalized mutual information* (NMI) and the *purity* metrics are used for clustering accuracy.

Table I shows that R-SAGMU provides better quality of PSRN than others. In addition, Table II reveals that R-SAGMU outperforms others with respect to the clustering quality.

TABLE I: Denoising quality comparison.

| Algorithm | PSNR |
|---|---|
| R-NMF | $23.867 \pm 0.168$ |
| ONMF | $11.208 \pm 0.008$ |
| R-ONMF | $23.452 \pm 0.101$ |
| R-SVRMU | $24.497 \pm 0.210$ |
| R-SAGMU | $\mathbf{25.205 \pm 0.124}$ |

TABLE II: Clustering accuracy comparison.

| Algorithm | NMI | Purity |
|---|---|---|
| R-NMF | $0.530 \pm 0.018$ | $0.403 \pm 0.023$ |
| ONMF | $0.246 \pm 0.011$ | $0.192 \pm 0.011$ |
| R-ONMF | $0.406 \pm 0.038$ | $0.322 \pm 0.028$ |
| R-SVRMU | $0.618 \pm 0.024$ | $0.508 \pm 0.030$ |
| R-SAGMU | $\mathbf{0.662 \pm 0.014}$ | $\mathbf{0.561 \pm 0.029}$ |

## VII. Conclusions

This present paper has proposed a novel accelerated stochastic multiplicative update with a gradient averaging technique: SAGMU. Numerical comparisons suggest that SAGMU robustly outperforms state-of-the-art algorithms across different synthetic and real-world datasets.

## Acknowledgements

H. Kasai was partially supported by JSPS KAKENHI Grant Numbers JP16K00031 and JP17H01732.

[2]http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html

## References

[1] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2001.
[2] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.
[3] R. Hibi and N. Takahashi, "A modified multiplicative update algorithm for euclidean distance-based nonnegative matrix factorization and its global convergence," in *ICONIP*, 2011.
[4] C.-J. Lin, "Projected gradient methods for non-negative matrix factorization," *Neural Computing*, vol. 19, no. 10, pp. 2756–2779, 2007.
[5] A. Cichocki and P. Anh-Huy, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92, no. 3, pp. 708–721, 2009.
[6] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural Computation*, vol. 24, no. 4, pp. 1085–1105, 2012.
[7] J. Kim, Y. He, and H. Park, "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework," *Journal of Global Optimization*, vol. 58, no. 2, pp. 285–319, 2014.
[8] S. S. Bucak and B. Gunsel, "Incremental subspace learning via nonnegative matrix factorization," *Pattern Recognition*, vol. 42, no. 5, pp. 788–797, 2009.
[9] A. Lefèvre, F. Bach, and C. Févotte, "Online algorithms for nonnegative matrix factorization with the itakura-saito divergence," in *WASPAA*, 2011.
[10] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Transations on Neural Network Learning Systems*, vol. 23, no. 7, pp. 1087–1099, 2012.
[11] R. Serizel, S. Essid, and G. Richard, "Mini-batch stochastic approaches for accelerated multiplicative updates in nonnegative matrix factorisation with beta-divergence," in *MLSP*, 2016.
[12] R. Zhao, V. Y. F. Tan, and H. Xu, "Online nonnegative matrix factorization with general divergences," in *AISTATS*, 2017.
[13] R. Zhao and V. Y. F. Tan, "Online nonnegative matrix factorization with outliers," in *ICASSP*, 2016.
[14] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.
[15] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization mehtods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.
[16] H. Kasai, "Stochastic variance reduced multiplicative update for nonnegative matrix factorization," in *ICASSP*, 2018.
[17] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *NIPS*, 2013.
[18] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *NIPS*, 2012.
[19] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, pp. 567–599, 2013.
[20] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *NIPS*, 2014.
[21] Y. Zhang and L. Xiao, "Stochastic primal-dual coordinate method for regularized empirical risk minimization," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.
[22] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takac, "SARAH: A novel method for machine learning problems using stochastic recursive gradient," in *ICML*, 2017.
[23] H. Jin, N. Feiping, and D. Chris, "Robust manifold nonnegative matrix factrization," *ACM Transations on Knowledge Discovery from Data*, vol. 8, no. 3, pp. 1–21, 2014.
[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.
[25] L. Bottou, "Online algorithm and stochastic approximations," in *On-Line Learning in Neural Networks*, D. Saad, Ed. Cambridge University Press, 1998.